<u>Question No. 15</u>

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#define Quantum 2
struct process
{
int at,copy_bt,bt,pr,pno,st,ct,position;
bool flag;
};

int comp(const void *a,const void *b)
{
struct process *ia=(struct process *)a;
struct process *ib=(struct process *)b;
if(ia->at == ib->at)
{
   return ia->pr-ib->pr;
}
else
{
   return ia->at-ib->at;
}
}

int pr_comp(const void *a,const void *b)
{
struct process *ia=(struct process *)a;
struct process *ib=(struct process *)b;
if(ia->pr == ib->pr)
{
   return ia->at-ib->at;
}
else
{
   return ia->pr-ib->pr;
}
}

int comp2(const void *a,const void *b)
{
struct process *ia=(struct process *)a;
struct process *ib=(struct process *)b;
return ia->position-ib->position;
}
```

```
void priority_st_ct_time(struct process Q[],int n,int *t)
{
    int pos=0;
    for(int i=0;i<n;i++)
    {
        Q[i].flag=false;
    }
    *t=Q[0].at;
    Q[0].st=*t;
    int j=1,k=0,count=0;
    for((*t)++;*t<=Q[n-1].at;(*t)++)
    {
        Q[k].bt--;
        count++;
        if(*t==Q[j].at)
        {
            if(Q[j].pr<Q[k].pr)
            {
                count=0;
                Q[k].flag=true;
                Q[k].position=pos++;
                k=j;
                Q[k].st=*t;
            }
            j++;
        }
        if(count==Quantum || Q[k].bt<=0)
        {
            if(Q[k].bt==0)
            {
                Q[k].ct=*t-1;
            }
            count=0;
            Q[k].flag=true;
            Q[k].position=pos++;
            int min=Q[k+1].pr,b=k+1;
            for(int a=k+2;a<j;a++)
            {
                if(Q[a].pr<min){
                    min=Q[a].pr;
                    b=a;
                }
            }
            k=b;
            Q[k].st=*t;
            j=k+1;
        }
    }
    qsort(Q,n,sizeof(struct process),pr_comp);
    k=0;
    int count1=0;
    for(;count1<n;(*t)++)
    {
        if(Q[k].flag)
        {
            k++;
            (*t)--;
            count1++;
            continue;
        }
        Q[k].bt--;
        count++;
        if(count==Quantum || Q[k].bt<=0)
        {
            count=0;count1++;
            if(Q[k].bt==0)
            {
                Q[k].ct=*t-1;
                Q[k].st=*t-Q[k].copy_bt;
            }
            else
            {
                Q[k].st=*t-Quantum;
            }
            Q[k].flag=true;
            Q[k].position=pos++;
            k++;
        }
    }
}

void round_robin_st_ct_time(struct process Q[],int n,int *t)
{
    qsort(Q,n,sizeof(struct process),comp2);
    while (1)
    {
        bool done = true;
        for (int i = 0 ; i < n; i++)
        {
            if (Q[i].bt > 0)
            {
                done = false;
                if (Q[i].bt > Quantum)
                {
                    *t += Quantum;
                    Q[i].bt -= Quantum;
                }
                else
                {
                    *t+=Q[i].bt;
                Q[i].ct=*t-1;
                Q[i].bt = 0;
              }
           }
        }
        if (done == true)
          break;
    }
}

void get_wt_time(struct process Q[],int tat[],int wt[],int n)
{
    for(int i=0;i<n;i++)
    wt[i]=tat[i]-Q[i].copy_bt;
}

void get_tat_time(struct process Q[],int tat[],int n)
{
for(int i=0;i<n;i++)
    tat[i]=Q[i].ct-Q[i].at;
}

void findgc(struct process Q[],int n)
{
int wt[n],tat[n],t=0;
double wavg=0,tavg=0;

//Fixed priority preemptive scheduling
priority_st_ct_time(Q,n,&t);

//round robin scheduling
round_robin_st_ct_time(Q,n,&t);

get_tat_time(Q,tat,n);
get_wt_time(Q,tat,wt,n);

printf("Process_no\tStart_time\tComplete_time\tTurn_Around_Time\tWaiting_Time\n");

for(int i=0;i<n;i++)
    {
        wavg += wt[i];
        tavg += tat[i];

        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",Q[i].pno,Q[i].st,Q[i].ct,tat[i],wt[i]);
    }

    printf("Average waiting time is : %f\n",wavg/(float)n);
    printf("average turnaround time : %f\n",tavg/(float)n);
}
```

```
int main()
{
   int n;
   printf("Enter the no. of process count : ");
   scanf("%d",&n);
   struct process Q1[n];
   printf("Enter the Arrival_time, Burst_time & Priority : e.g. (4 7 2)\n");
   for(int i=0;i<n;i++)
   {
      Q1[i].pno=i+1;
      printf("For Process %d : ",Q1[i].pno);
      scanf("%d %d %d",&Q1[i].at,&Q1[i].bt,&Q1[i].pr);
      Q1[i].copy_bt=Q1[i].bt;
   }
   qsort(Q1,n,sizeof(struct process),comp);
   findgc(Q1,n);
   return 0;
}
```